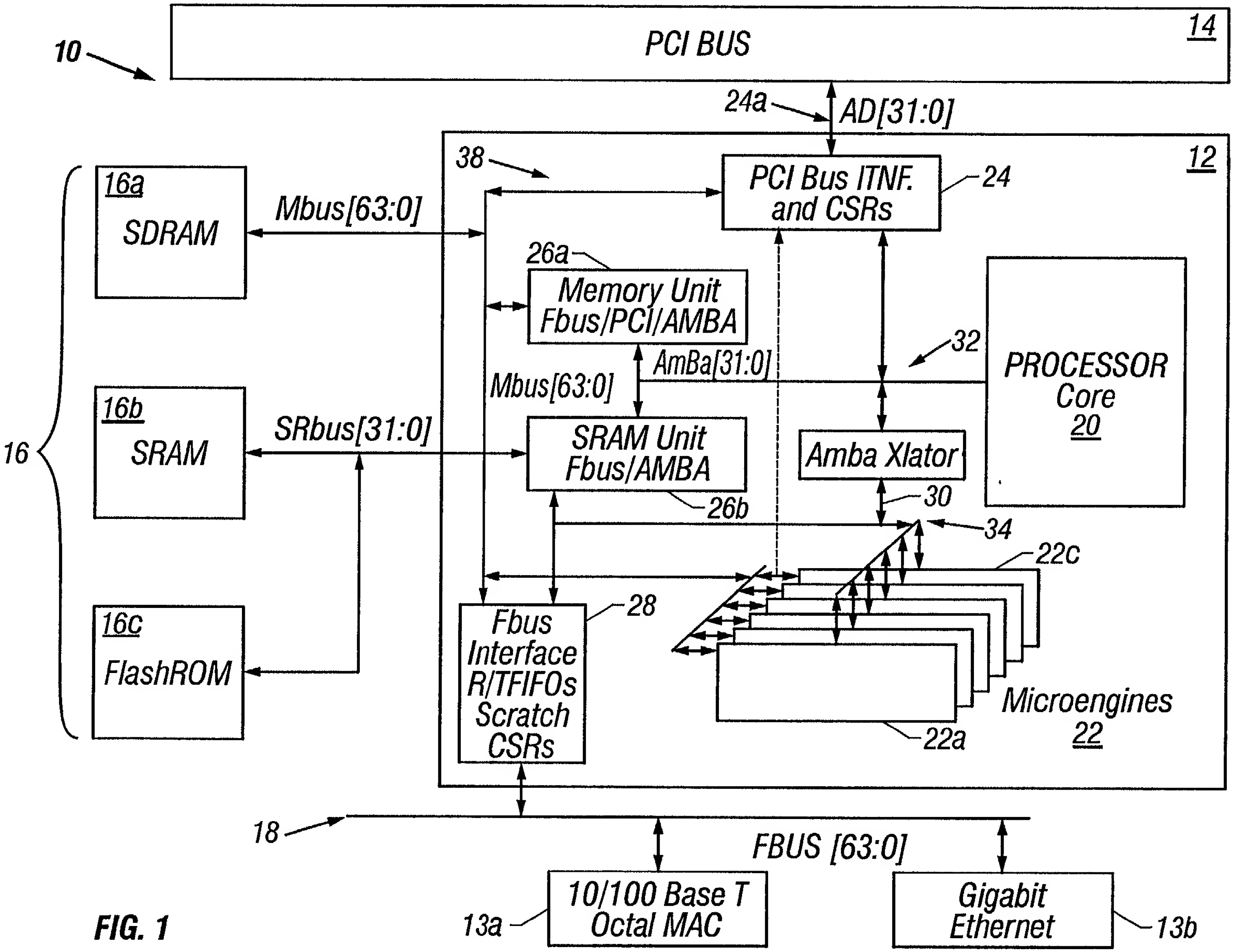

10
PCI BUS
14
24a
AD[31:0]
12
38
PCI Bus ITNF. and CSRs
24
16a
SDRAM
Mbus[63:0]
26a
Memory Unit Fbus/PCI/AMBA
32
PROCESSOR Core 20
AmBa[31:0]
Mbus[63:0]
16
16b
SRAM
SRbus[31:0]
SRAM Unit Fbus/AMBA
Amba Xlator
26b
30
34
22c
16c
FlashROM
28
Fbus Interface R/TFIFOs Scratch CSRs
Microengines 22
22a
18
FBUS [63:0]
10/100 Base T Octal MAC
13a
Gigabit Ethernet
13b

FIG. 1

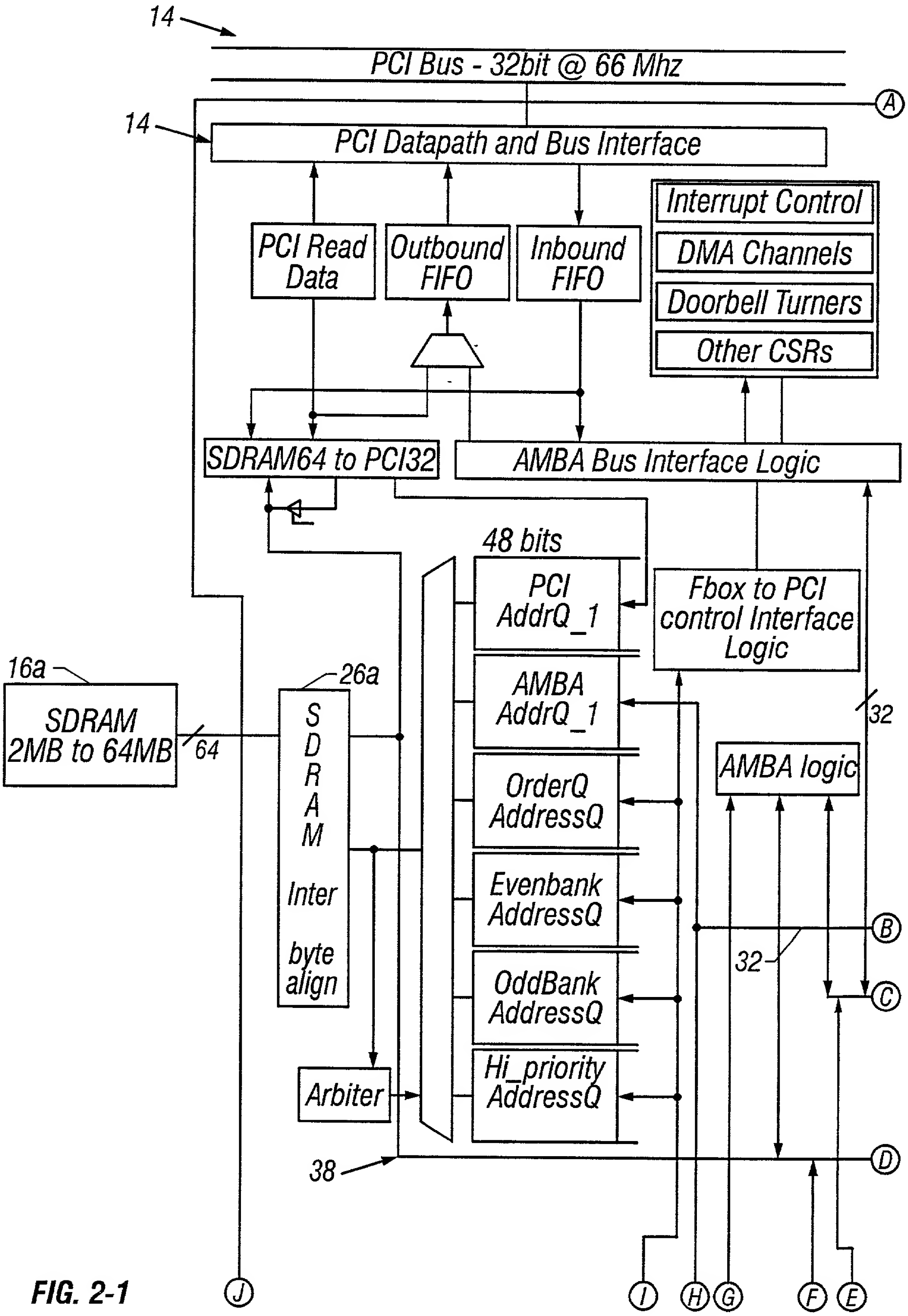

14
PCI Bus - 32bit @ 66 Mhz
A
14
PCI Datapath and Bus Interface
PCI Read Data
Outbound FIFO
Inbound FIFO
Interrupt Control
DMA Channels
Doorbell Turners
Other CSRs
SDRAM64 to PCI32
AMBA Bus Interface Logic
48 bits
PCI AddrQ_1
Fbox to PCI control Interface Logic
16a
SDRAM 2MB to 64MB
64
26a
S D R A M Inter byte align
AMBA AddrQ_1
32
OrderQ AddressQ
AMBA logic
Evenbank AddressQ
B
32
OddBank AddressQ
C
Arbiter
Hi_priority AddressQ
38
D
J
I
H
G
F
E

FIG. 2-1

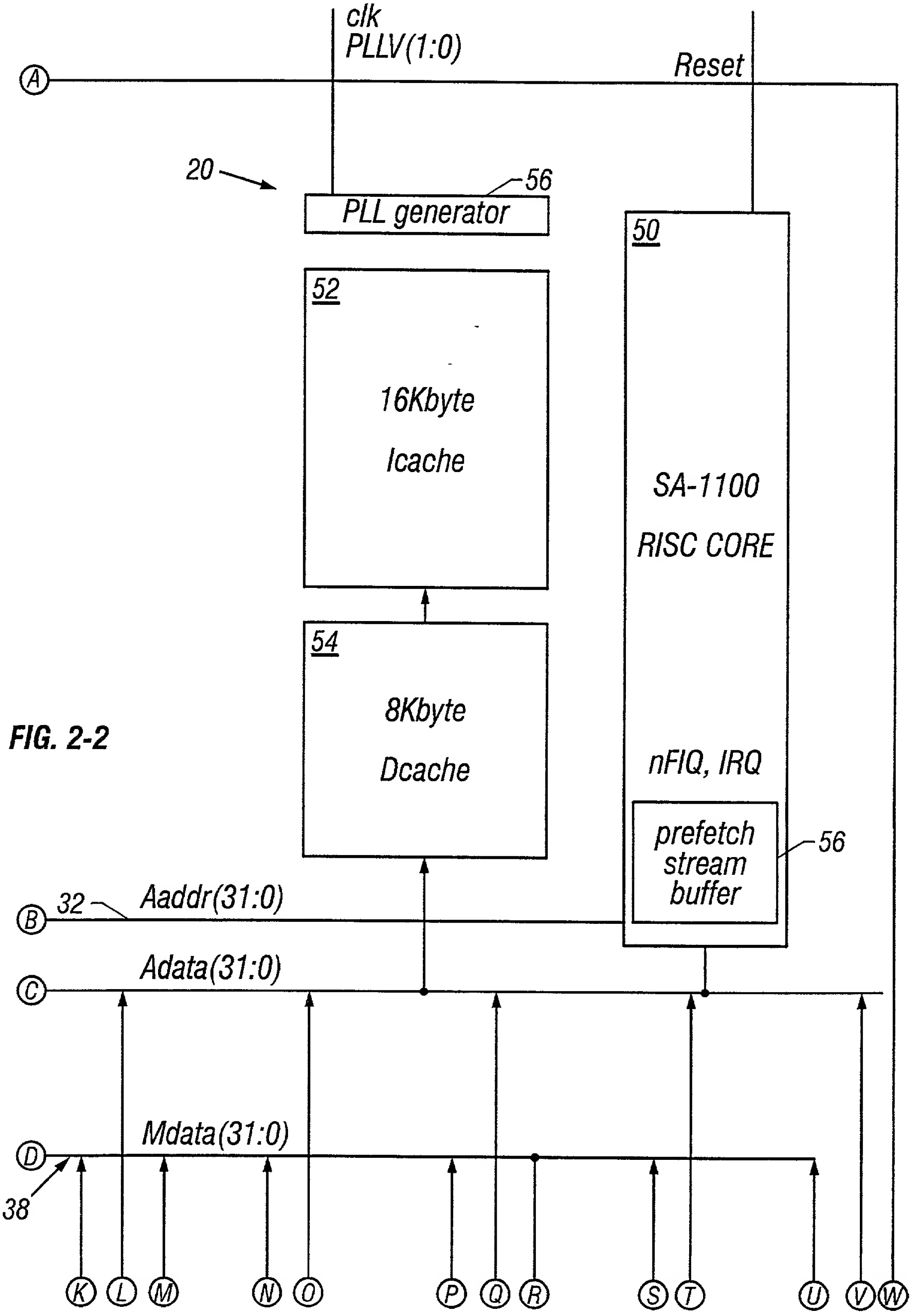

clk
PLLV(1:0)
Reset
A
20
56
PLL generator
50
52
16Kbyte
Icache
SA-1100
RISC CORE
54
8Kbyte
Dcache
nFIQ, IRQ
prefetch
stream
buffer
56
B
32
Aaddr(31:0)
C
Adata(31:0)
D
Mdata(31:0)
38
K
L
M
N
O
P
Q
R
S
T
U
V
W

FIG. 2-2

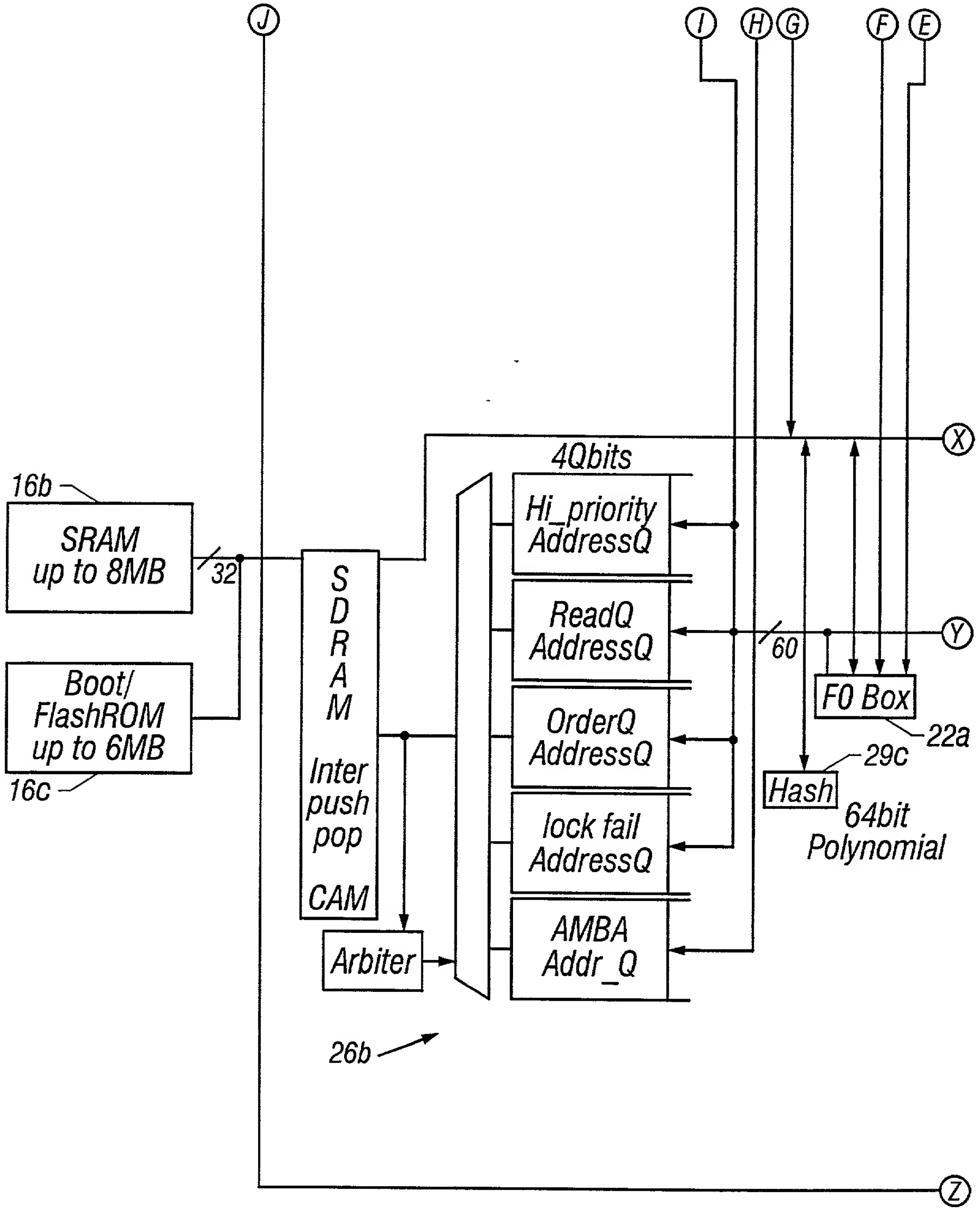

J
I
H
G
F
E
X
Y
Z
16b
SRAM up to 8MB
32
Boot/ FlashROM up to 6MB
16c
S D R A M Inter push pop CAM
Arbiter
4Qbits
Hi_priority AddressQ
ReadQ AddressQ
OrderQ AddressQ
lock fail AddressQ
AMBA Addr_Q
60
F0 Box
22a
29c
Hash
64bit Polynomial
26b

FIG. 2-3

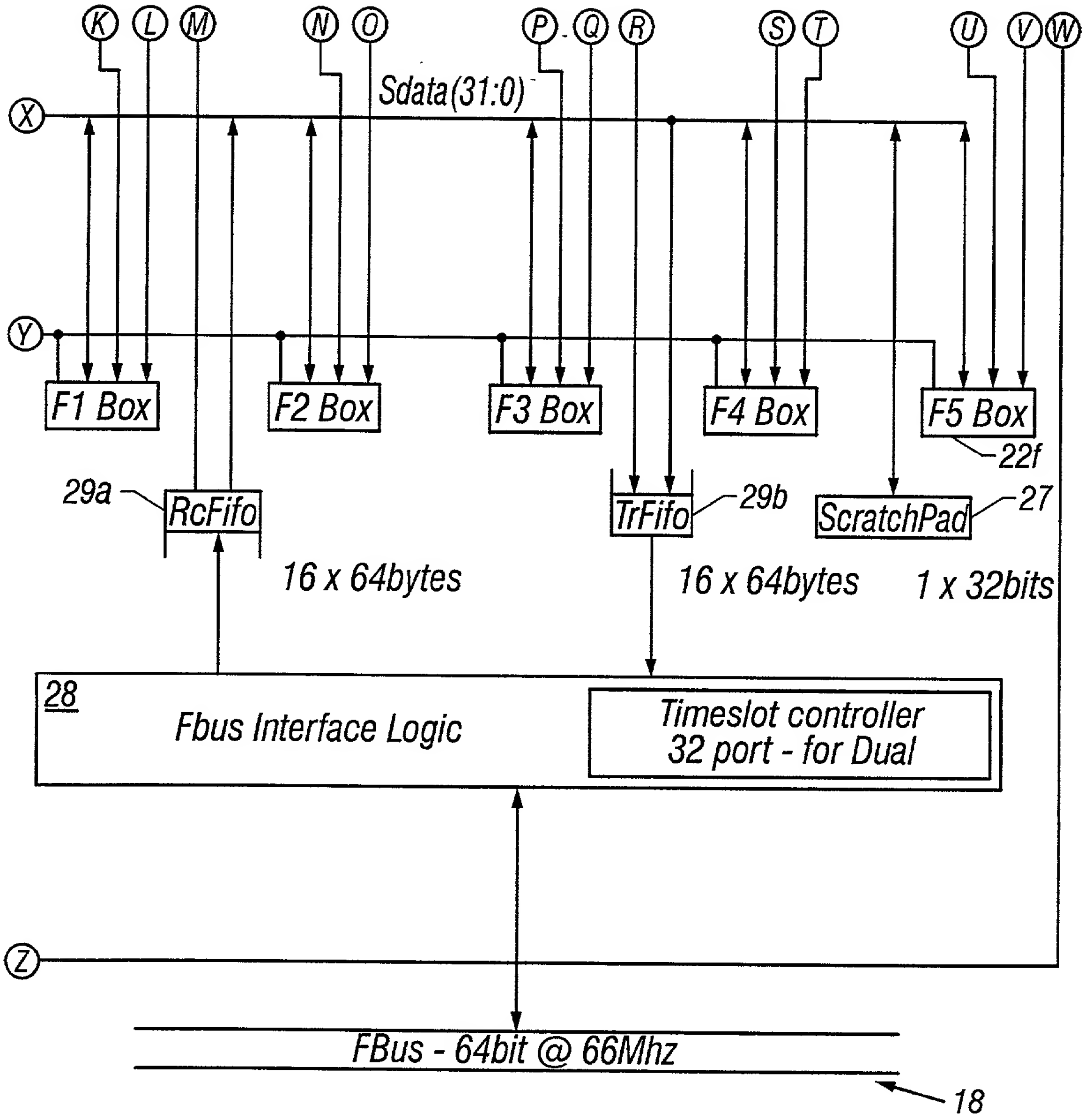

K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Sdata(31:0)
Y
F1 Box
F2 Box
F3 Box
F4 Box
F5 Box
22f
29a
RcFifo
29b
TrFifo
ScratchPad
27
16 x 64bytes
16 x 64bytes
1 x 32bits
28
Fbus Interface Logic
Timeslot controller
32 port - for Dual
Z
FBus - 64bit @ 66Mhz
18

FIG. 2-4

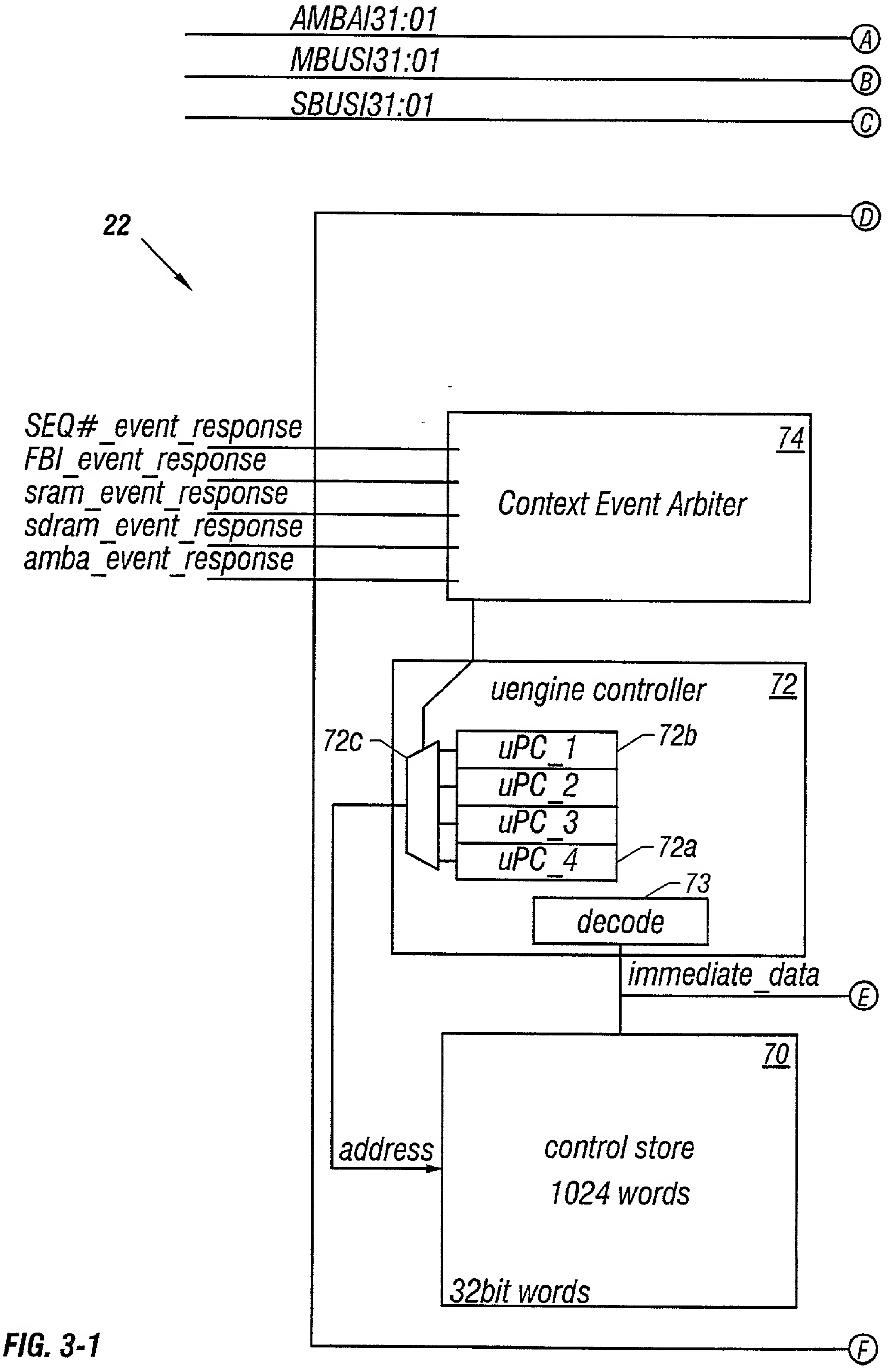

AMBAI31:01
MBUSI31:01
SBUSI31:01
A
B
C
D
22
SEQ#_event_response
FBI_event_response
sram_event_response
sdram_event_response
amba_event_response
74
Context Event Arbiter
uengine controller
72
72c
uPC_1
uPC_2
uPC_3
uPC_4
72b
72a
73
decode
immediate_data
E
70
address
control store
1024 words
32bit words
F

FIG. 3-1

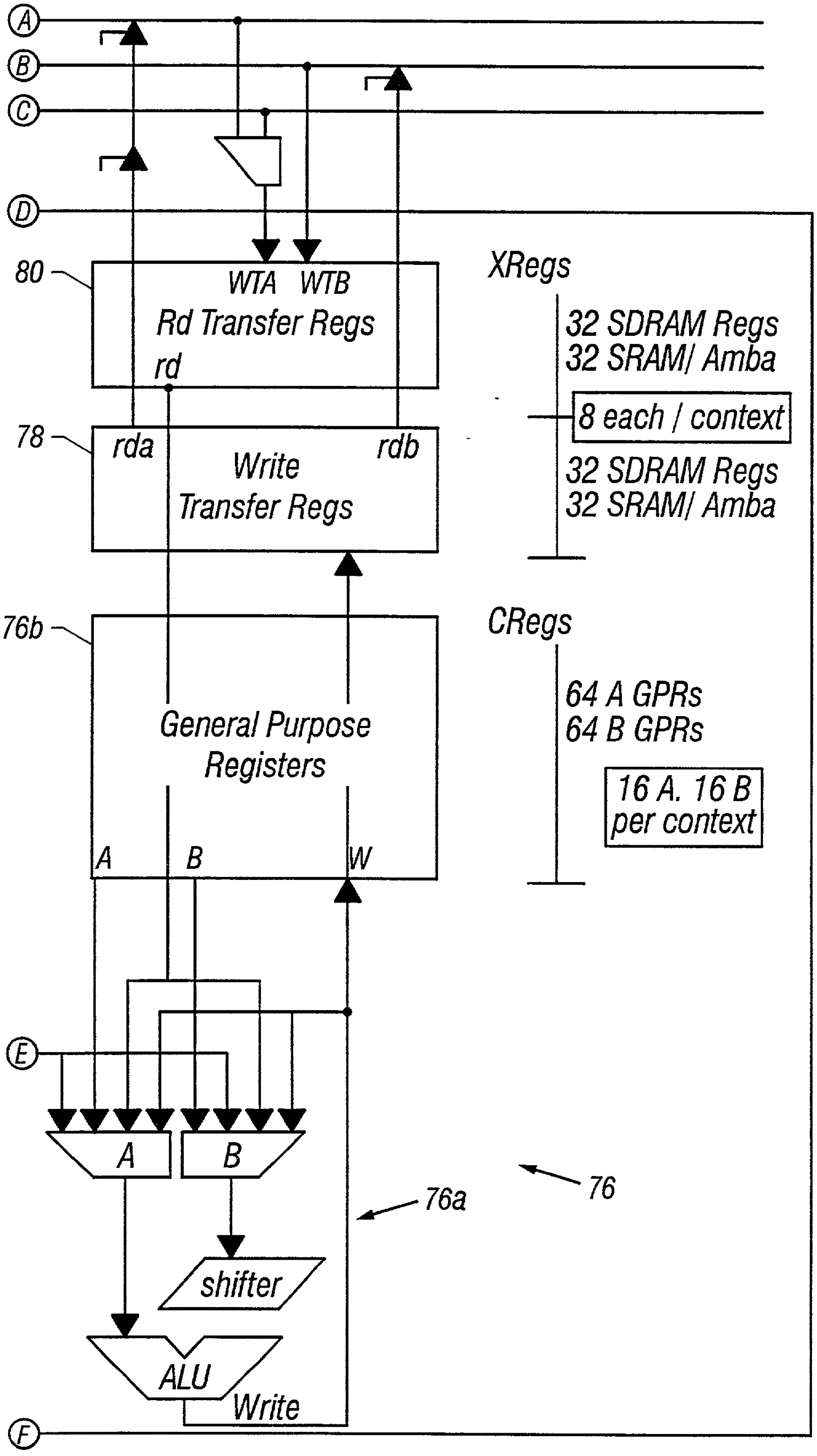

A
B
C
D
80
WTA WTB
Rd Transfer Regs
rd
78
rda
rdb
Write
Transfer Regs
XRegs
32 SDRAM Regs
32 SRAM/ Amba
8 each / context
32 SDRAM Regs
32 SRAM/ Amba
76b
General Purpose
Registers
CRegs
64 A GPRs
64 B GPRs
16 A. 16 B
per context
A
B
W
E
A
B
shifter
ALU
Write
F
76a
76

FIG. 3-2

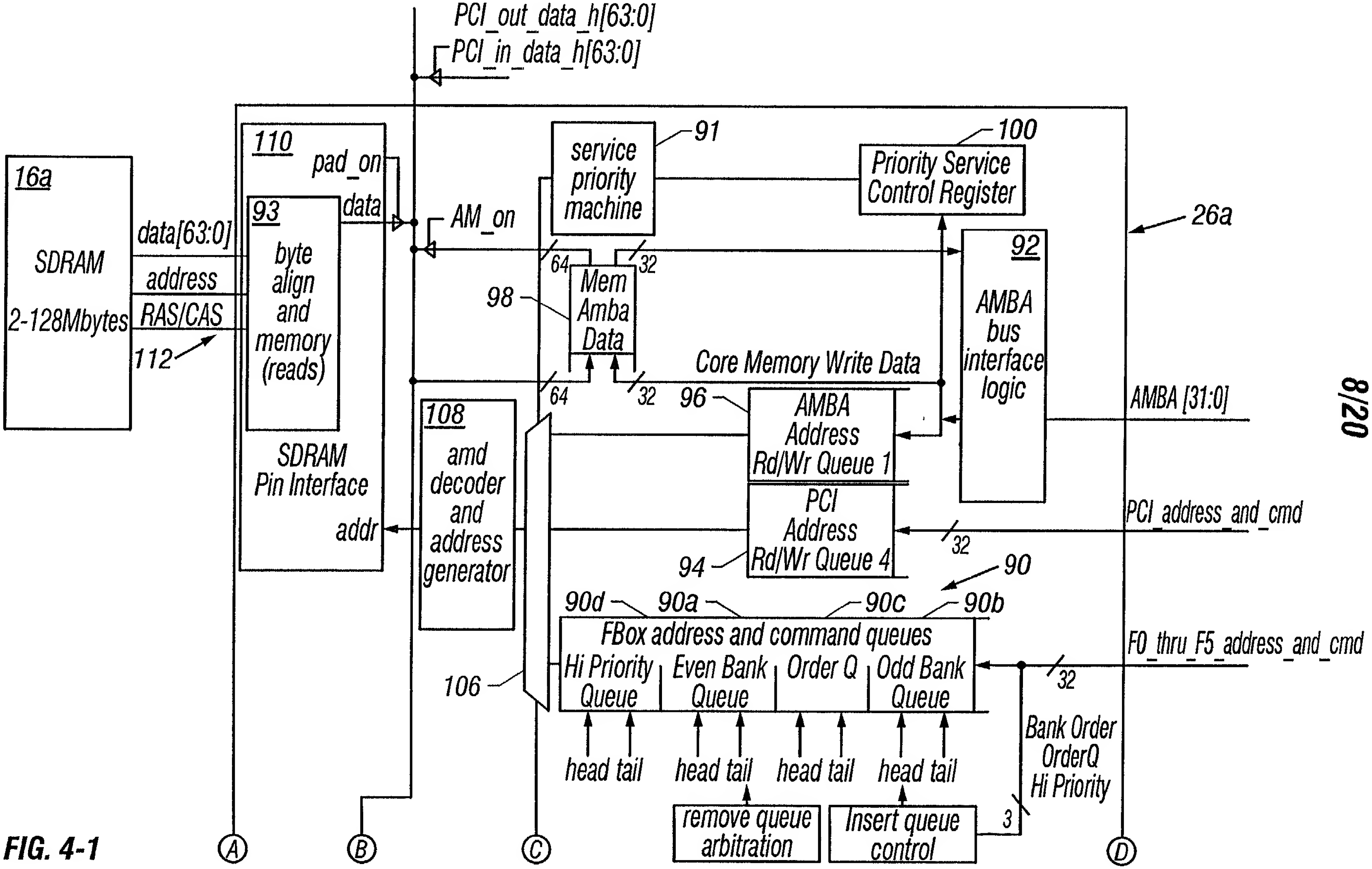

PCI_out_data_h[63:0]
PCI_in_data_h[63:0]
16a
SDRAM
2-128Mbytes
data[63:0]
address
RAS/CAS
112
110
pad_on
93
data
byte align and memory (reads)
SDRAM Pin Interface
addr
AM_on
service priority machine
91
Priority Service Control Register
100
26a
64
32
Mem Amba Data
98
92
AMBA bus interface logic
64
32
Core Memory Write Data
96
AMBA Address Rd/Wr Queue 1
AMBA [31:0]
108
amd decoder and address generator
PCI Address Rd/Wr Queue 4
PCI_address_and_cmd
32
94
90
90d
90a
90c
90b
FBox address and command queues
Hi Priority Queue
Even Bank Queue
Order Q
Odd Bank Queue
F0_thru_F5_address_and_cmd
32
106
head tail
head tail
head tail
head tail
Bank Order
OrderQ
Hi Priority
3
remove queue arbitration
Insert queue control
A
B
C
D

FIG. 4-1

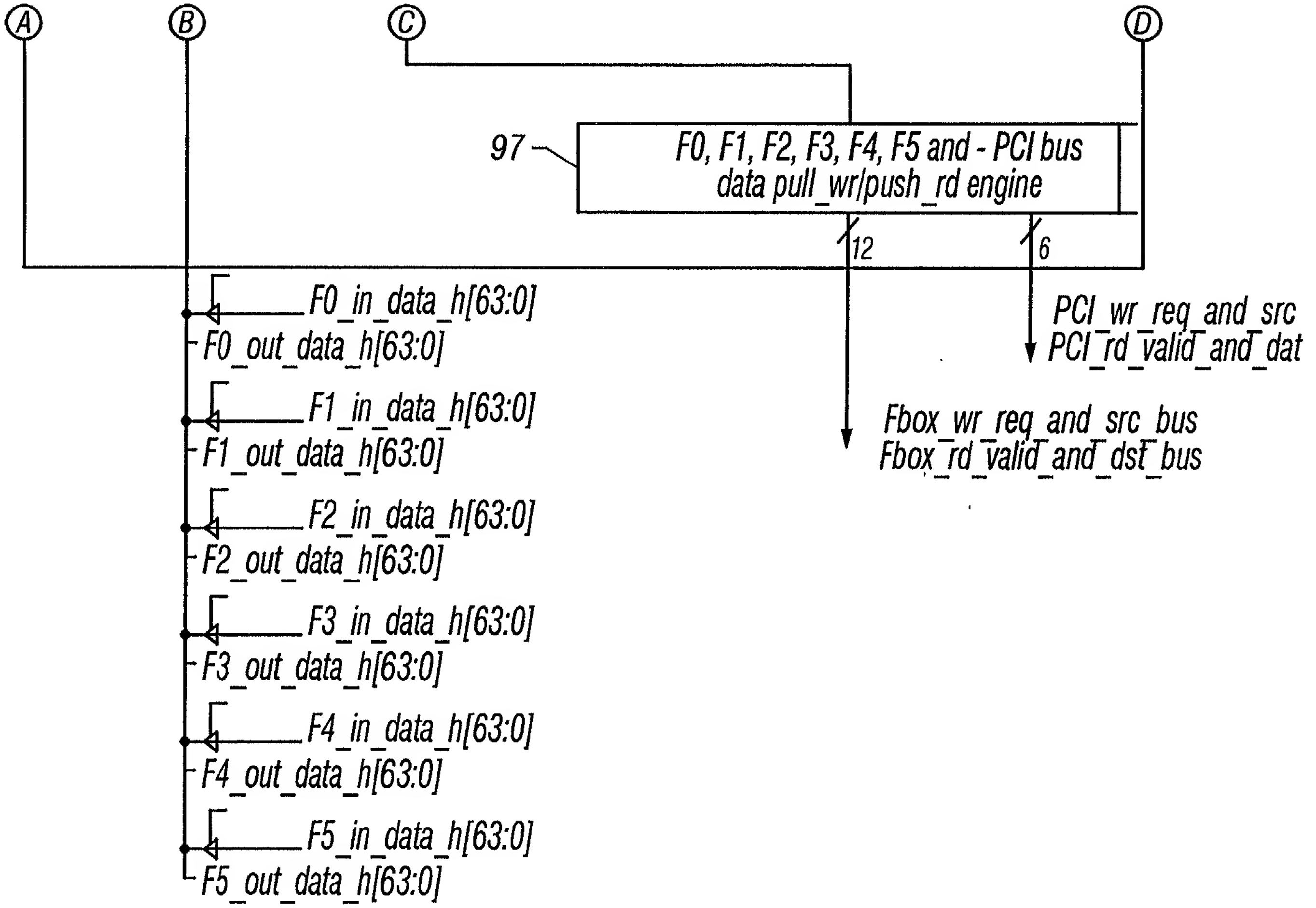

A
B
C
D
97
F0, F1, F2, F3, F4, F5 and - PCI bus data pull_wr/push_rd engine
12
6
PCI_wr_req_and_src
PCI_rd_valid_and_dat
Fbox_wr_req_and_src_bus
Fbox_rd_valid_and_dst_bus
F0_in_data_h[63:0]
F0_out_data_h[63:0]
F1_in_data_h[63:0]
F1_out_data_h[63:0]
F2_in_data_h[63:0]
F2_out_data_h[63:0]
F3_in_data_h[63:0]
F3_out_data_h[63:0]
F4_in_data_h[63:0]
F4_out_data_h[63:0]
F5_in_data_h[63:0]
F5_out_data_h[63:0]

FIG. 4-2

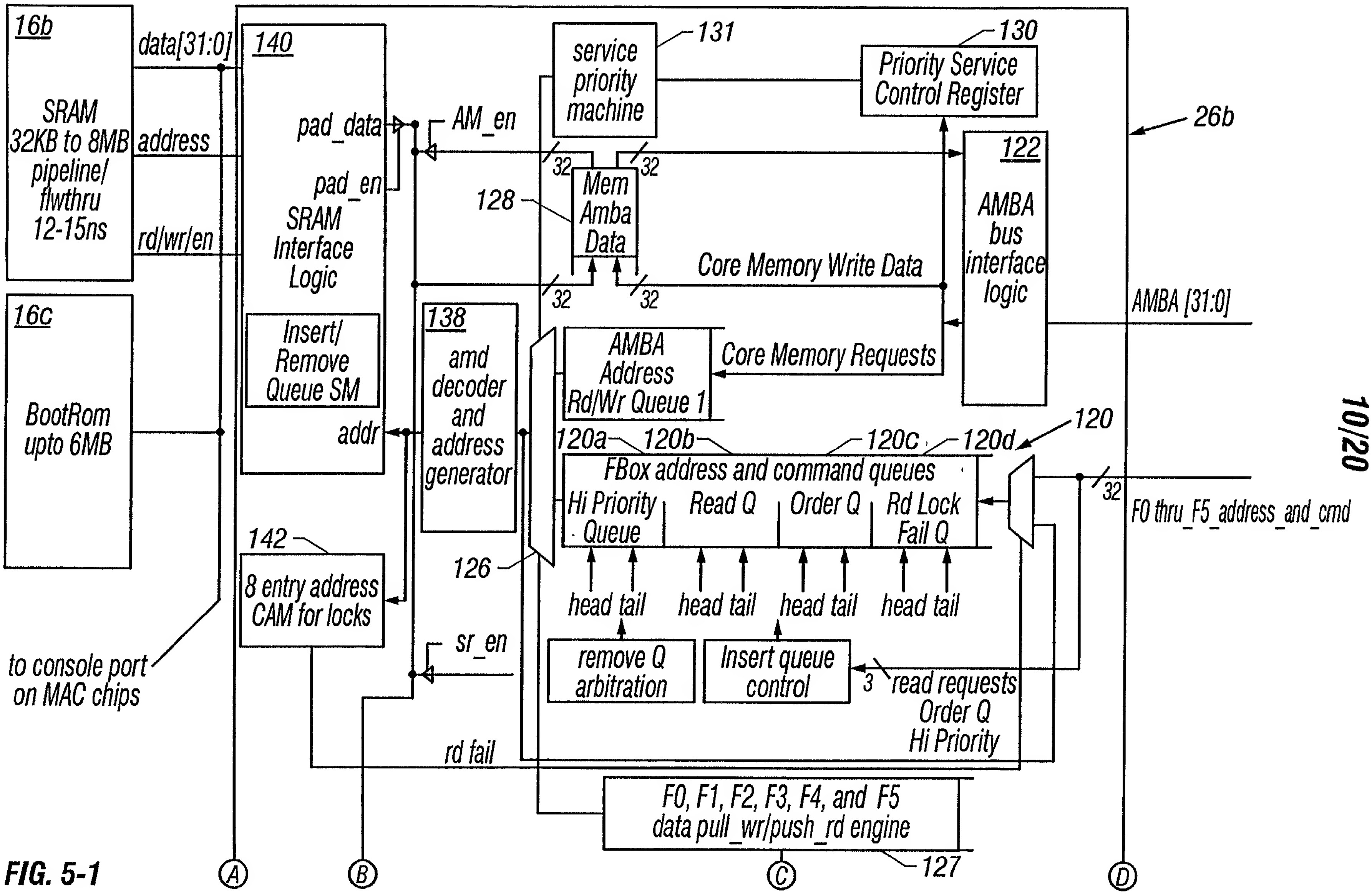
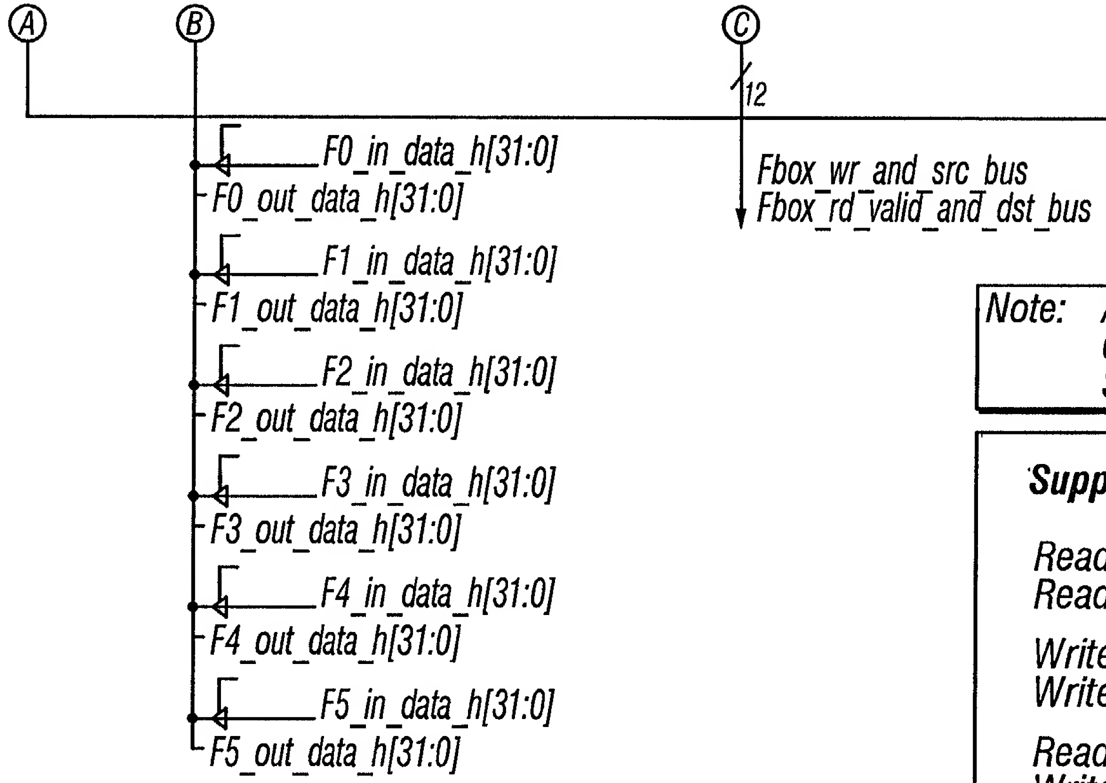

16b
SRAM 32KB to 8MB pipeline/ flwthru 12-15ns
data[31:0]
address
rd/wr/en
16c
BootRom upto 6MB
to console port on MAC chips
140
pad_data
pad_en
SRAM Interface Logic
Insert/ Remove Queue SM
addr
AM_en
142
8 entry address CAM for locks
sr_en
138
amd decoder and address generator
131
service priority machine
128
Mem Amba Data
32
32
32
32
130
Priority Service Control Register
122
AMBA bus interface logic
26b
Core Memory Write Data
Core Memory Requests
AMBA Address Rd/Wr Queue 1
AMBA [31:0]
120a
120b
120c
120d
120
FBox address and command queues
Hi Priority Queue
Read Q
Order Q
Rd Lock Fail Q
head tail
head tail
head tail
head tail
126
remove Q arbitration
Insert queue control
3
read requests Order Q Hi Priority
32
F0 thru_F5_address_and_cmd
rd fail
F0, F1, F2, F3, F4, and F5 data pull_wr/push_rd engine
127
A
B
C
D

FIG. 5-1



Note:	Address:	22 bits
	Comment:	4 bits
	Src/Dst:	6 bits

Supported Comments:

Read 1 questword
Read 8 questwords

Write 1 questword
Write 8 questwords

Read/Lock
Write/Unlock
Unlock
Reserve n locks

Insert Queue Element
Remove Queue Element

FIG. 5-2

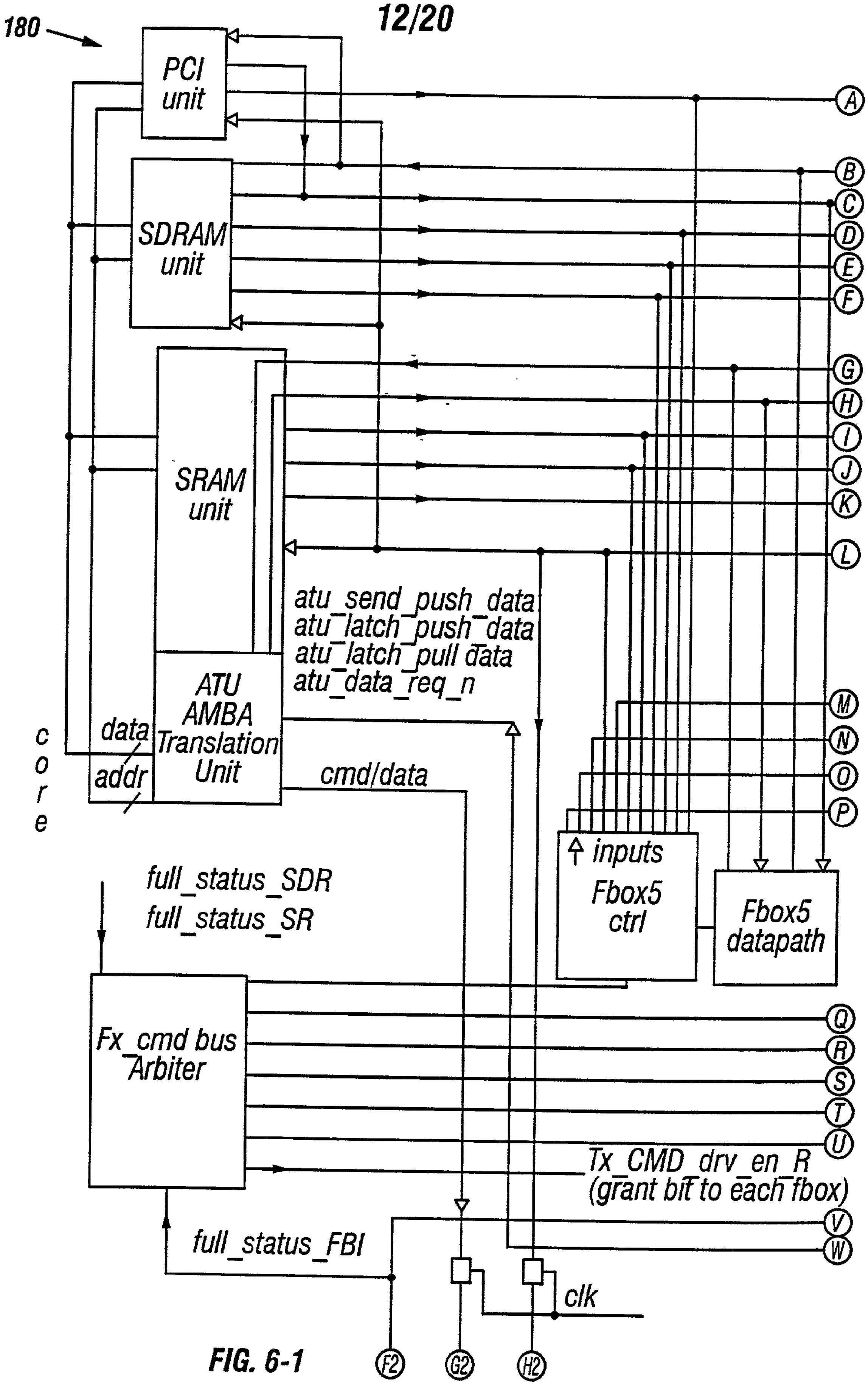

180
PCI unit
SDRAM unit
SRAM unit
ATU AMBA Translation Unit
data
addr
core
atu_send_push_data
atu_latch_push_data
atu_latch_pull data
atu_data_req_n
cmd/data
full_status_SDR
full_status_SR
inputs
Fbox5 ctrl
Fbox5 datapath
Fx_cmd bus Arbiter
Tx_CMD_drv_en_R
(grant bit to each fbox)
full_status_FBI
clk
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
F2
G2
H2

FIG. 6-1

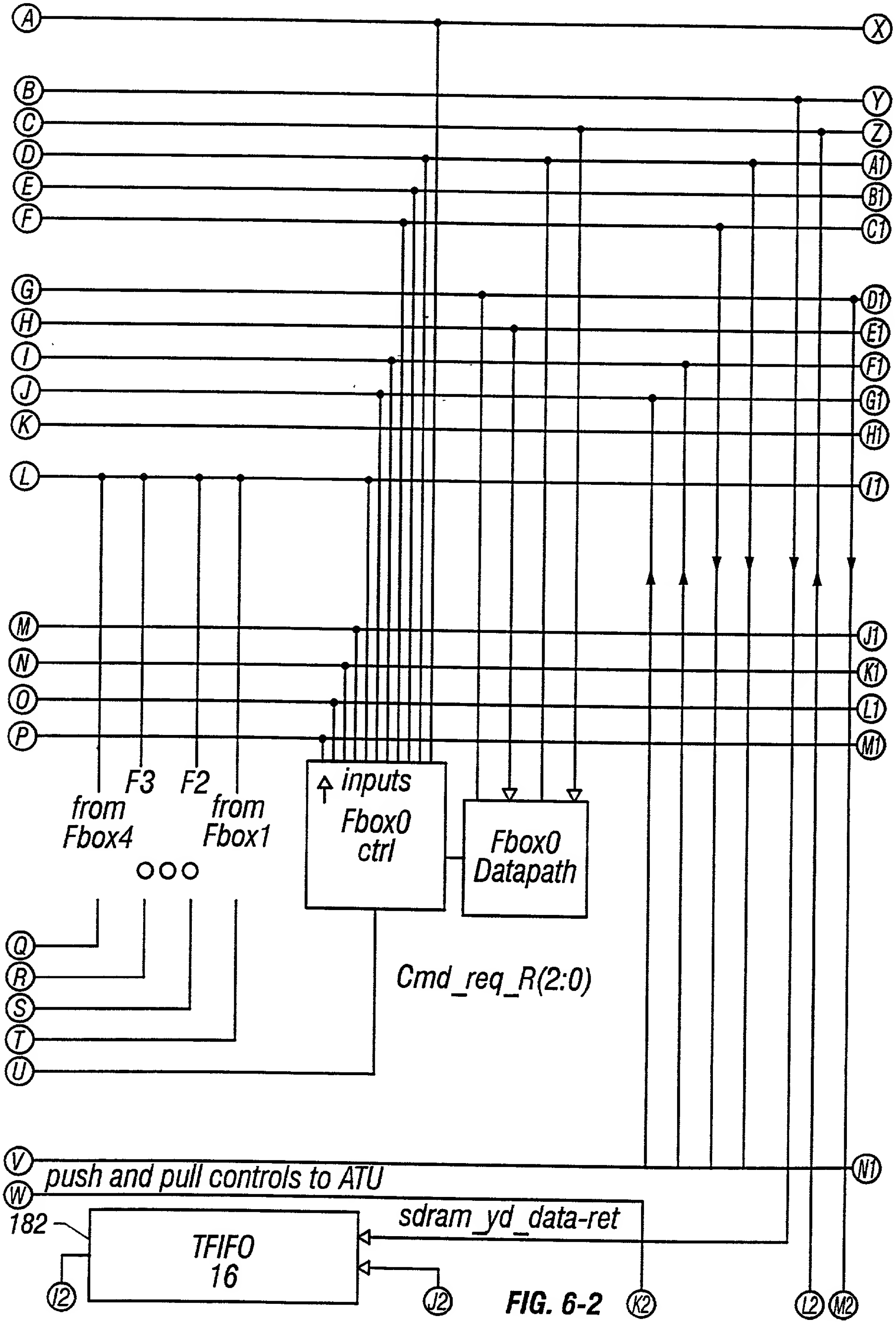

A
B
C
D
E
F
G
H
I
J
K
L
X
Y
Z
A1
B1
C1
D1
E1
F1
G1
H1
I1
M
N
O
P
J1
K1
L1
M1
F3
F2
from Fbox4
from Fbox1
inputs
Fbox0 ctrl
Fbox0 Datapath
Q
R
S
T
U
Cmd_req_R(2:0)
V
W
N1
push and pull controls to ATU
182
TFIFO 16
sdram_yd_data-ret
I2
J2
K2
L2
M2

FIG. 6-2

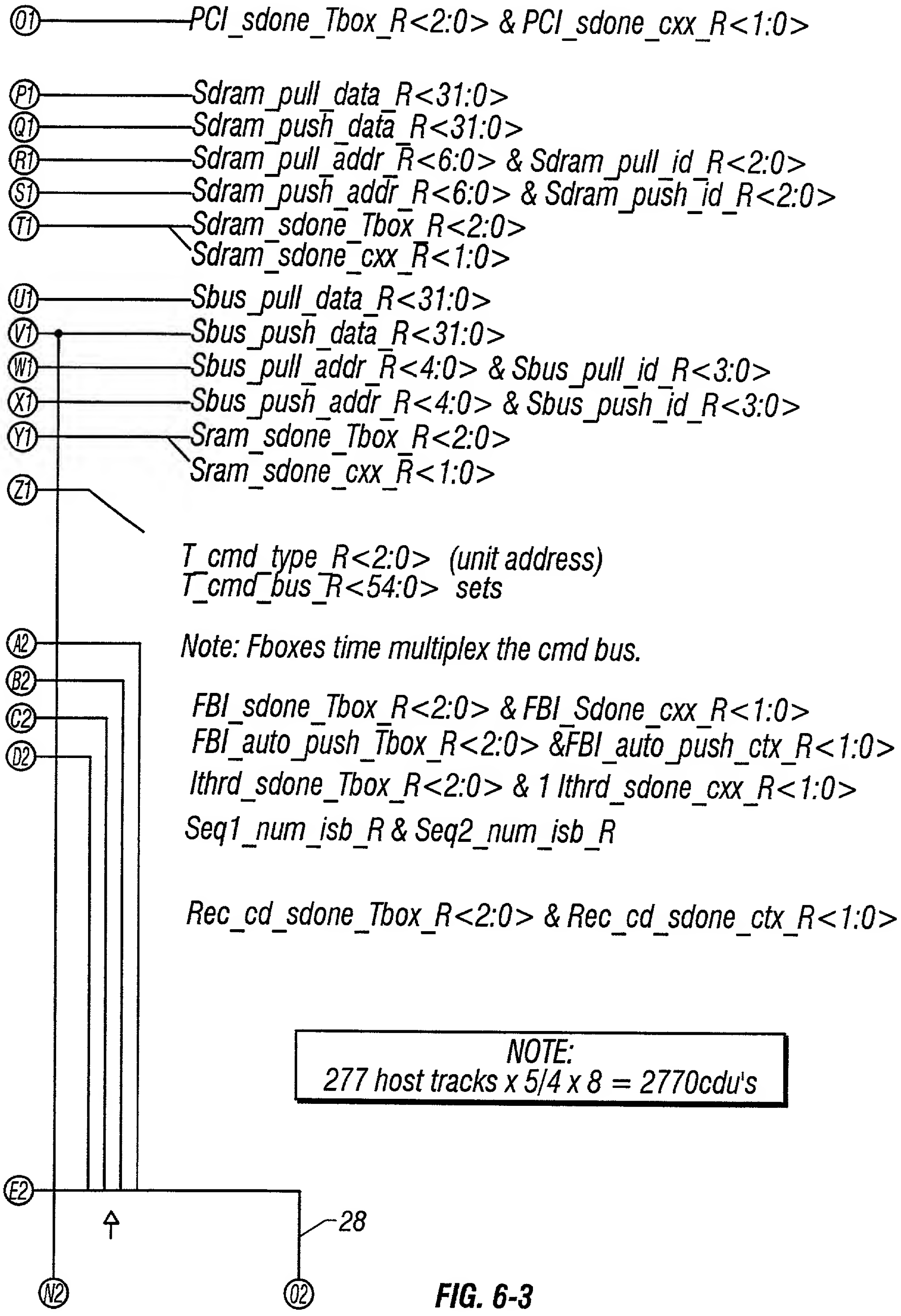

O1
PCI_sdone_Tbox_R<2:0> & PCI_sdone_cxx_R<1:0>
P1
Sdram_pull_data_R<31:0>
Q1
Sdram_push_data_R<31:0>
R1
Sdram_pull_addr_R<6:0> & Sdram_pull_id_R<2:0>
S1
Sdram_push_addr_R<6:0> & Sdram_push_id_R<2:0>
T1
Sdram_sdone_Tbox_R<2:0>
Sdram_sdone_cxx_R<1:0>
U1
Sbus_pull_data_R<31:0>
V1
Sbus_push_data_R<31:0>
W1
Sbus_pull_addr_R<4:0> & Sbus_pull_id_R<3:0>
X1
Sbus_push_addr_R<4:0> & Sbus_push_id_R<3:0>
Y1
Sram_sdone_Tbox_R<2:0>
Sram_sdone_cxx_R<1:0>
Z1
T_cmd_type_R<2:0> (unit address)
T_cmd_bus_R<54:0> sets
Note: Fboxes time multiplex the cmd bus.
A2
B2
C2
D2
FBI_sdone_Tbox_R<2:0> & FBI_Sdone_cxx_R<1:0>
FBI_auto_push_Tbox_R<2:0> &FBI_auto_push_ctx_R<1:0>
Ithrd_sdone_Tbox_R<2:0> & 1 Ithrd_sdone_cxx_R<1:0>
Seq1_num_isb_R & Seq2_num_isb_R
Rec_cd_sdone_Tbox_R<2:0> & Rec_cd_sdone_ctx_R<1:0>
NOTE:
277 host tracks x 5/4 x 8 = 2770cdu's
E2
28
N2
O2

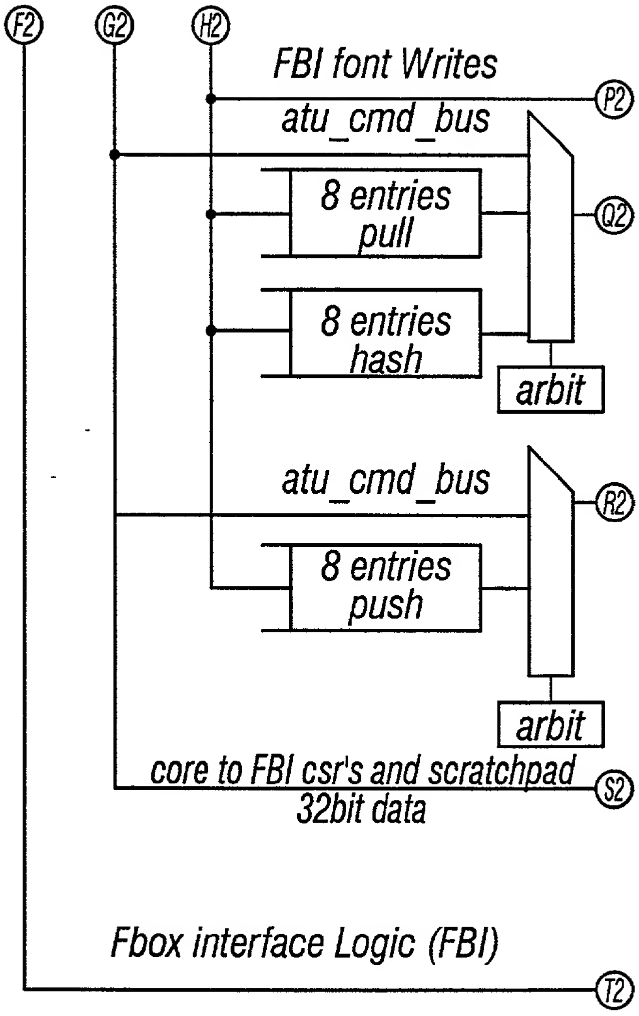
FIG. 6-3

Pull Engine
Cmd Arbitration

1) Amba
2) Hash
3) Pull Cmds

Pull Engine
Cmd Arbitration

1) Amba
2)Hash Completion
3)Push Cmds



ATU Notes:

a) Core to FboxRegs:
use sram_push_data_bus

b) Core to FBI Regs:
use private ATU/FBI
cmd/data bus

c) Core reads FboxRegs:
use SRAM_pull_data_bus

d) Core reads FBIRegs:
use sram_push_data_bus
(makes sram appear like
another Fbox to FBI on
sram_push_bus)

Cmd_Req_R<2:0>
000 none
001 Sram Chain
010 SDR chain
011 Sram
100 SDR
101 FBI
110 PCI
111
Tx_CMD_drv_en_R<1:0>
0 none
1 grant

FIG. 6-4

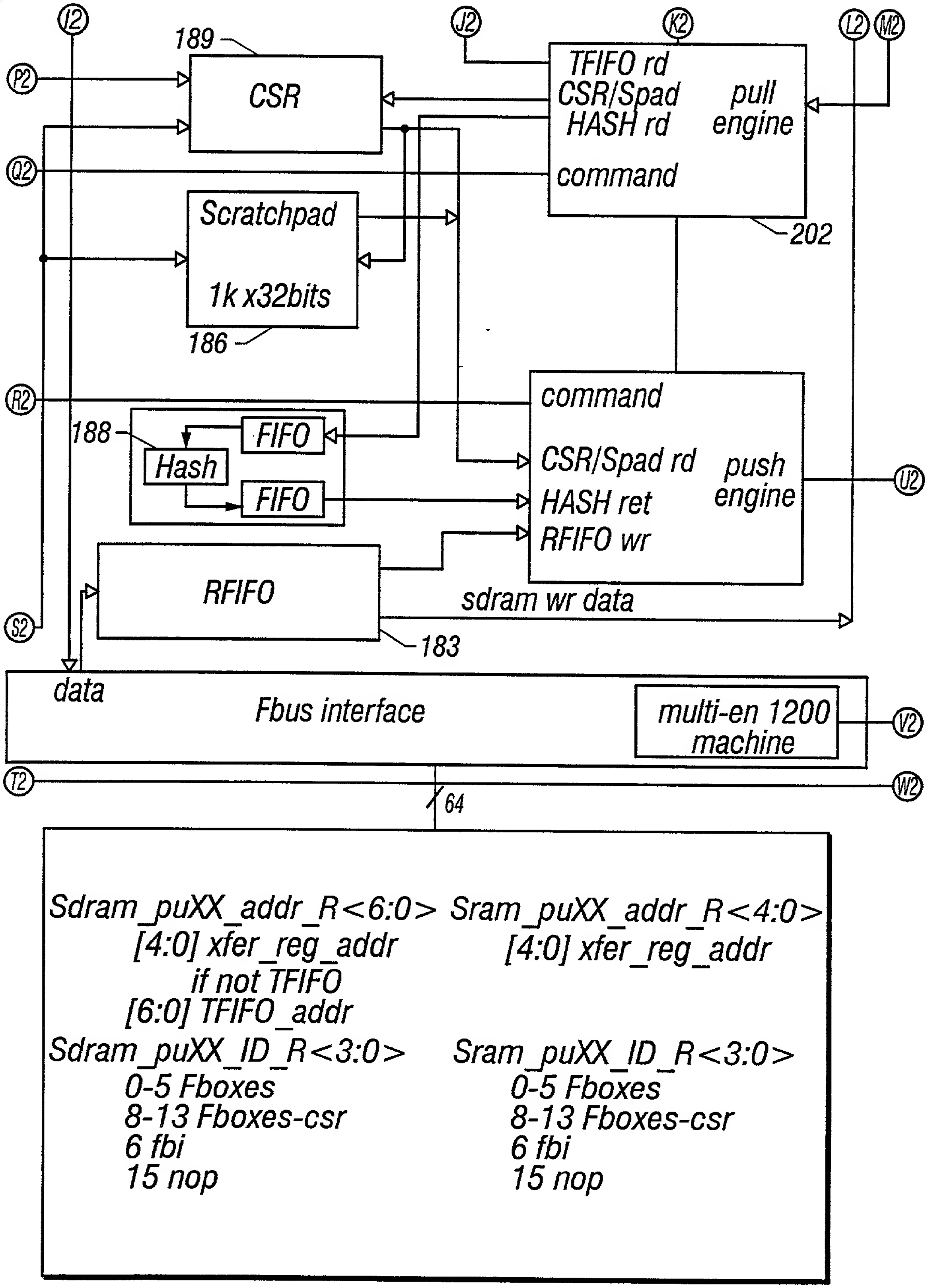
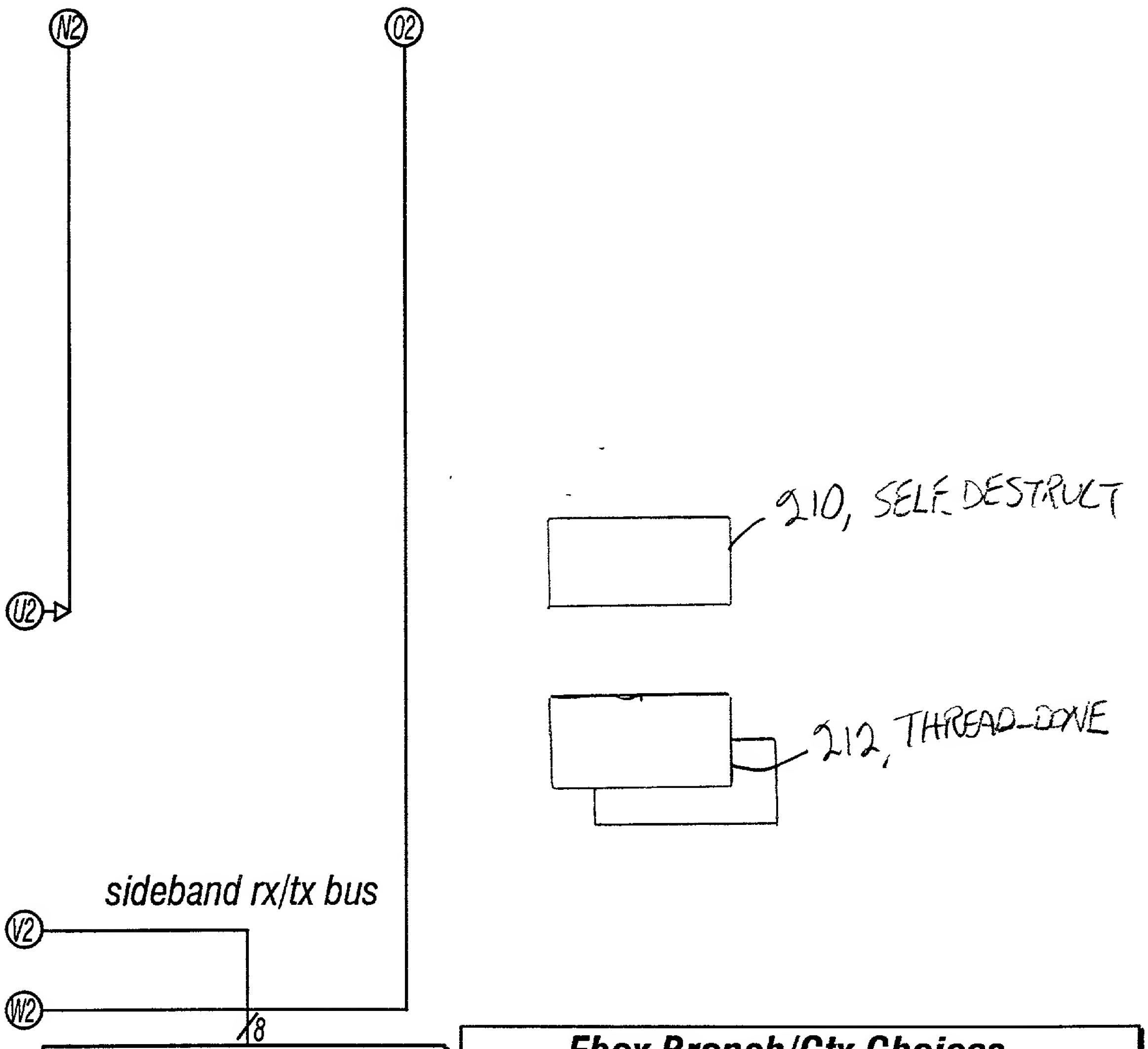

I2
189
P2
CSR
J2
K2
L2
M2
TFIFO rd
CSR/Spad
HASH rd
pull engine
command
Q2
Scratchpad
1k x32bits
202
186
R2
command
188
FIFO
Hash
FIFO
CSR/Spad rd
HASH ret
RFIFO wr
push engine
U2
RFIFO
sdram wr data
S2
183
data
Fbus interface
multi-en 1200 machine
V2
T2
W2
64
Sdram_puXX_addr_R<6:0>
[4:0] xfer_reg_addr
if not TFIFO
[6:0] TFIFO_addr
Sram_puXX_addr_R<4:0>
[4:0] xfer_reg_addr
Sdram_puXX_ID_R<3:0>
0-5 Fboxes
8-13 Fboxes-csr
6 fbi
15 nop
Sram_puXX_ID_R<3:0>
0-5 Fboxes
8-13 Fboxes-csr
6 fbi
15 nop

FIG. 6-5



T_Cmd_type_R<2:0>

000: bus idle
001: SDRAM
010: SRAM
011: SRAM-csr
100: PCI
101: reserved
110: FBI
111: Scratch

Fbox Branch/Ctx Choices

1) FBI_sdone		br / ctx
2) FBI_auto_push		br / ctx
3) Ithread_sdone		br / ctx
4) signal_rec_cxt		br / ctx
5) Seq#1_change	(flag)	br / ctx
6) Seq#2 change	(flag)	br / ctx
7)SRAM_sdone		br / ctx
8)SDRAM_sdone		br / ctx
9) volunteer_cxx_swap		ctx
10) Rec_req_available	(flag)	br
11)SDRAM rd parity en	(flag)	br
12) Fbox_push_protect		br
13) ccodes, contexts and kill		

FIG. 6-6

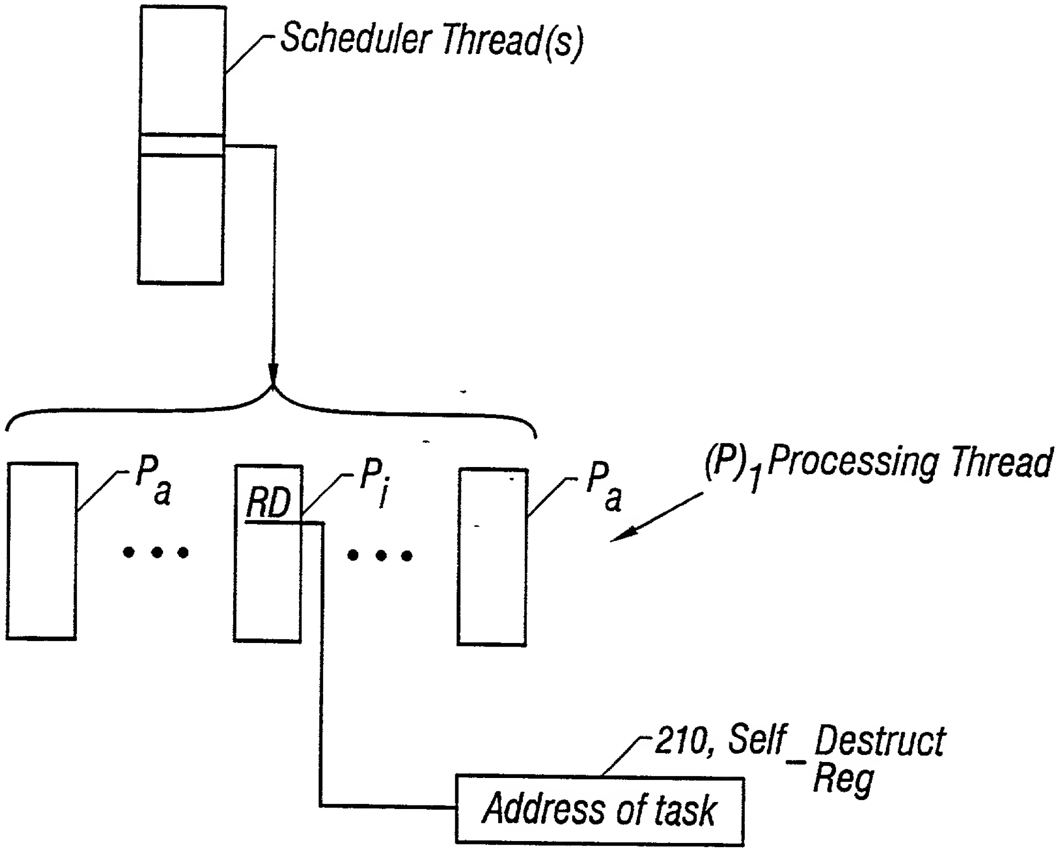

Scheduler Thread(s)
Pa
RD
Pi
Pa
(P)1 Processing Thread
210, Self_Destruct Reg
Address of task

FIG. 7A

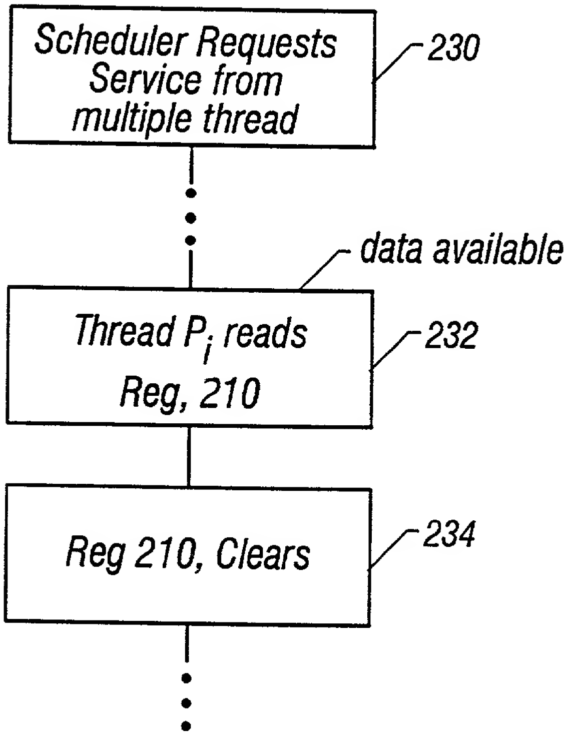

Scheduler Requests Service from multiple thread
230
data available
Thread Pi reads Reg, 210
232
Reg 210, Clears
234

FIG. 7B

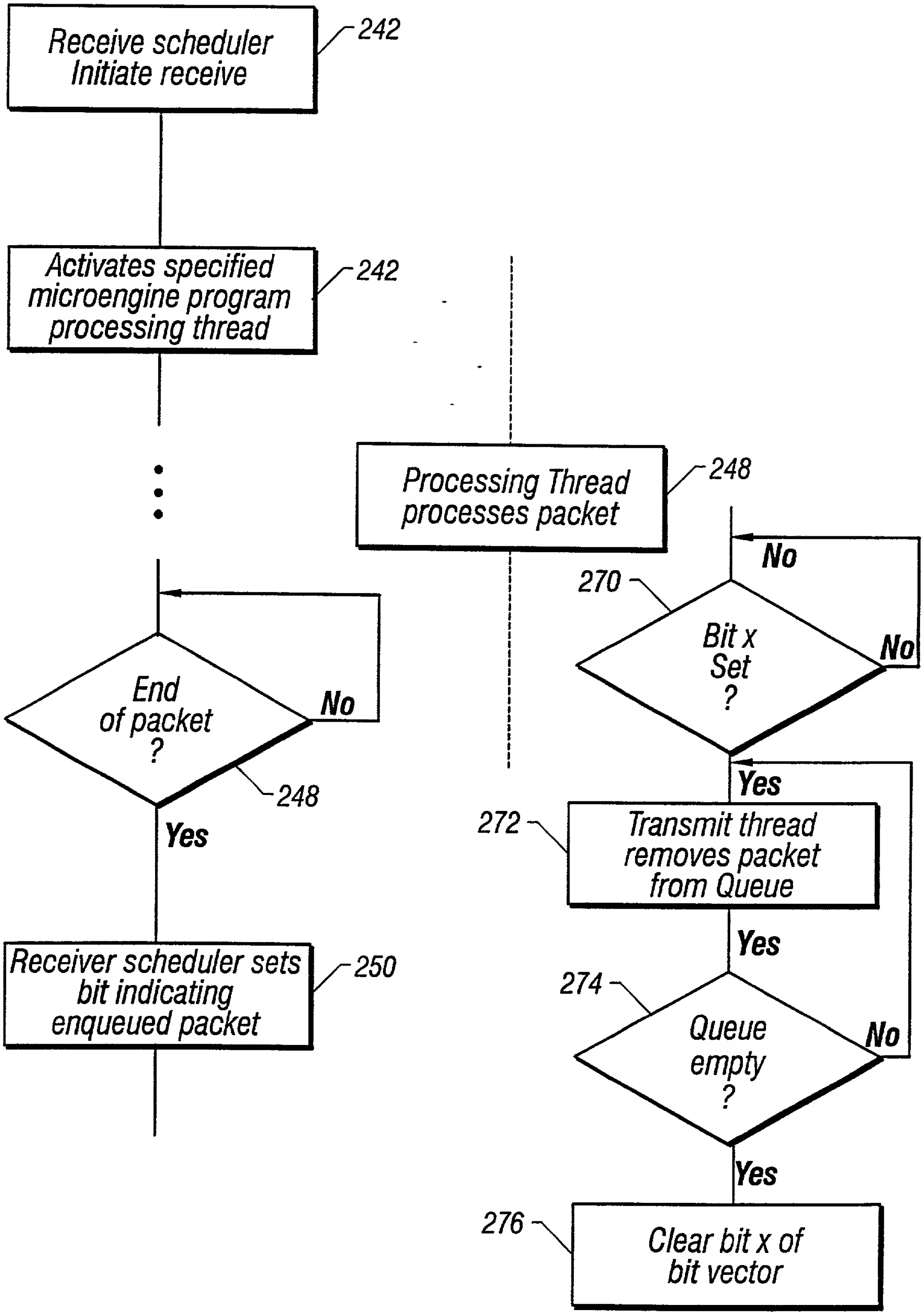

Receive scheduler
Initiate receive
242
Activates specified microengine program processing thread
242
Processing Thread processes packet
248
End of packet ?
No
248
Yes
Receiver scheduler sets bit indicating enqueued packet
250
No
270
Bit x Set ?
No
Yes
272
Transmit thread removes packet from Queue
Yes
274
Queue empty ?
No
Yes
276
Clear bit x of bit vector

FIG. 8

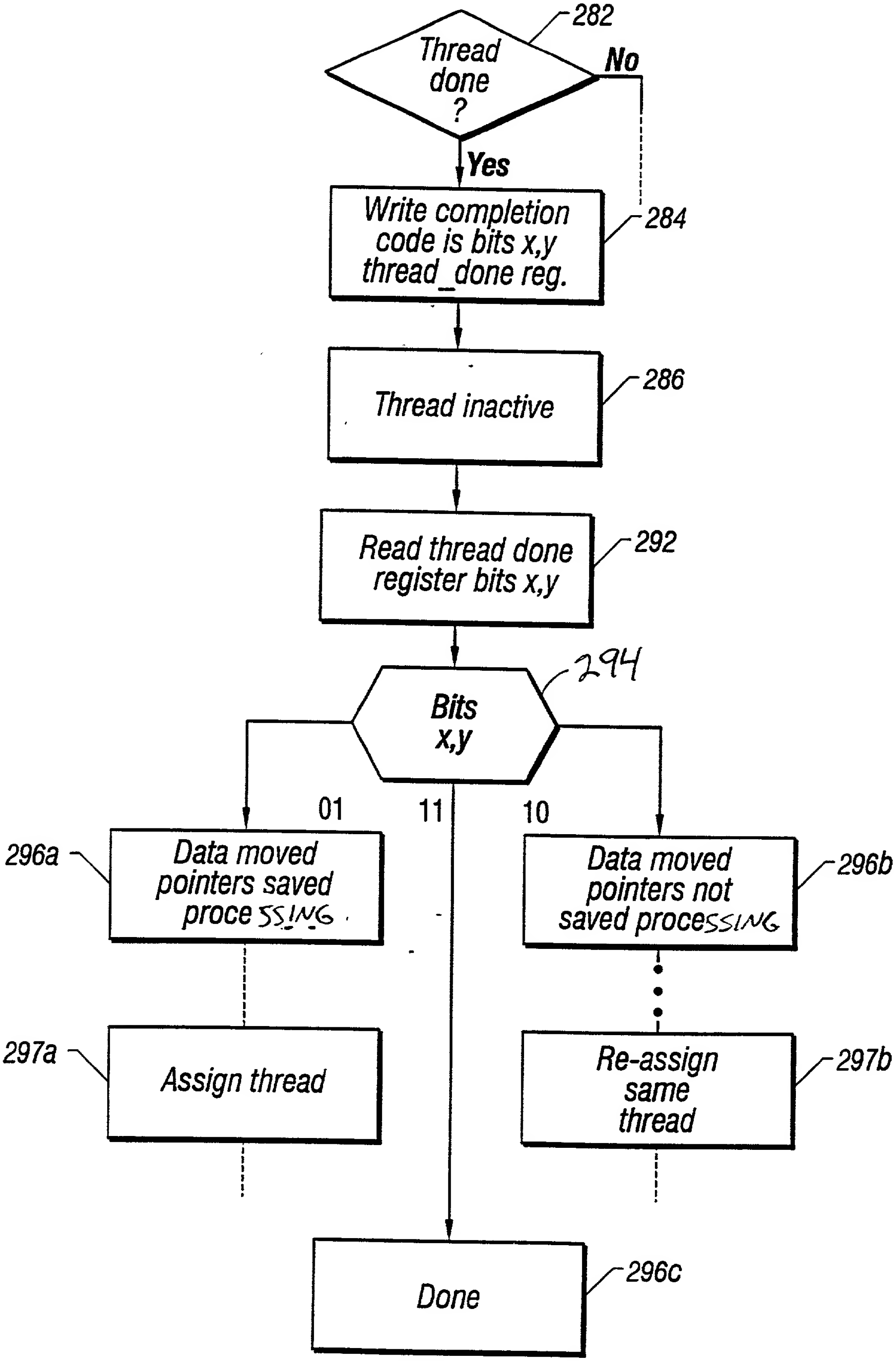

Thread done ?
282
No
Yes
Write completion code is bits x,y thread_done reg.
284
Thread inactive
286
Read thread done register bits x,y
292
Bits x,y
294
01
11
10
Data moved pointers saved processing
296a
Data moved pointers not saved processing
296b
Assign thread
297a
Re-assign same thread
297b
Done
296c

FIG 9